MOi workshop on Deep Learning

October 2022

Ronan Fablet
Prof. IMT Atlantique
INRIA Team Odyssey

Daria Botvinko
PhD student
ENIB/IMT Atlantique
INRIA Team Odyssey

Quentin Febvre
Phd student
IMT Atlantique
INRIA Team Odyssey

Objectives

Key objective at the end of the workshop: ability to deploy a deep learning approach for Moi-related topics

Content :

- Introduction of the main Deep learning concepts
- Introduction to learning paradigms for real-world MOi topics
- Introduction to PyTorch Deep Learning “ecosystem”
- Training through practice (labsession and project session)

Overview of the course

	Matin 9h30-12h30	Après-midi 14h00-17h30
Jour 1	Introduction au Deep Learning et à la librairie Pytroch	Séance projet #1
Jour 2	Réseaux convolutifs	Introduction à Pytorch Lightning, Séance projet #2
Jour 3	Auto-encodeurs et Réseaux récurrents	Introduction à Tensorboard Séance projet #3
Jour 4	Deep Learning, PINN (Physics-informed neural networks) et Assimilation de Données	Séance projet #4, Présentation courte des projets
Jour 5	Introduction à hydra Séance projet #5 + Restitution des projets	

Resources

- **Book: Deep Learning**

Goodfellow, Bengio, Courville, MIT Press

Online version http://www.deeplearningbook.org/

- **Online course by Andrew Ng (Stanford/Baidu)**

  Youtube: [link]

  Online course on Coursera: [link]

- **Review paper: Deep learning in neural networks by**

  J. Schmidhube pdf: [link]

- **Github repo: https://github.com/CIA-Oceanix/DLCourse_MOi_2022**

Ready to go ?

- **Access to discord server ?**

- **Google cola account ?**

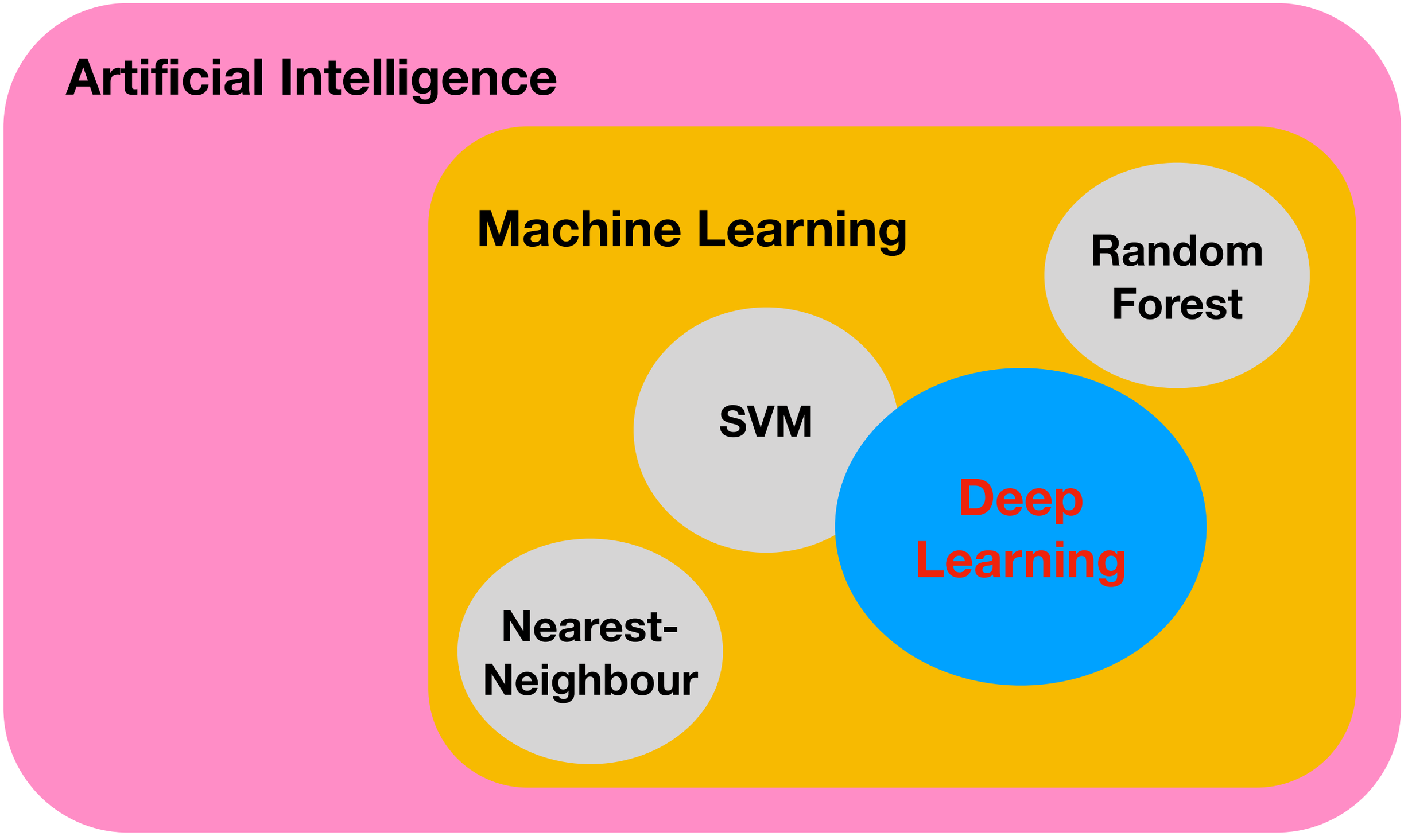
Course #1: Introduction to Deep Learning

Roadmap

- What is deep learning ?
- What does (machine) learning mean ?
- What are Neural Networks ?
- How to implement DL models using PyTorch ?
- Guidelines to implement a deep learning approach

What is Deep Learning?

What is Deep Learning?



Deep learning has rapidly become the state-of-art-framework for a wide range of applications in computer vision, natural language processing, signal processing…

From Image to Text

(image captioning)

A woman is throwing a **frisbee** in a park.

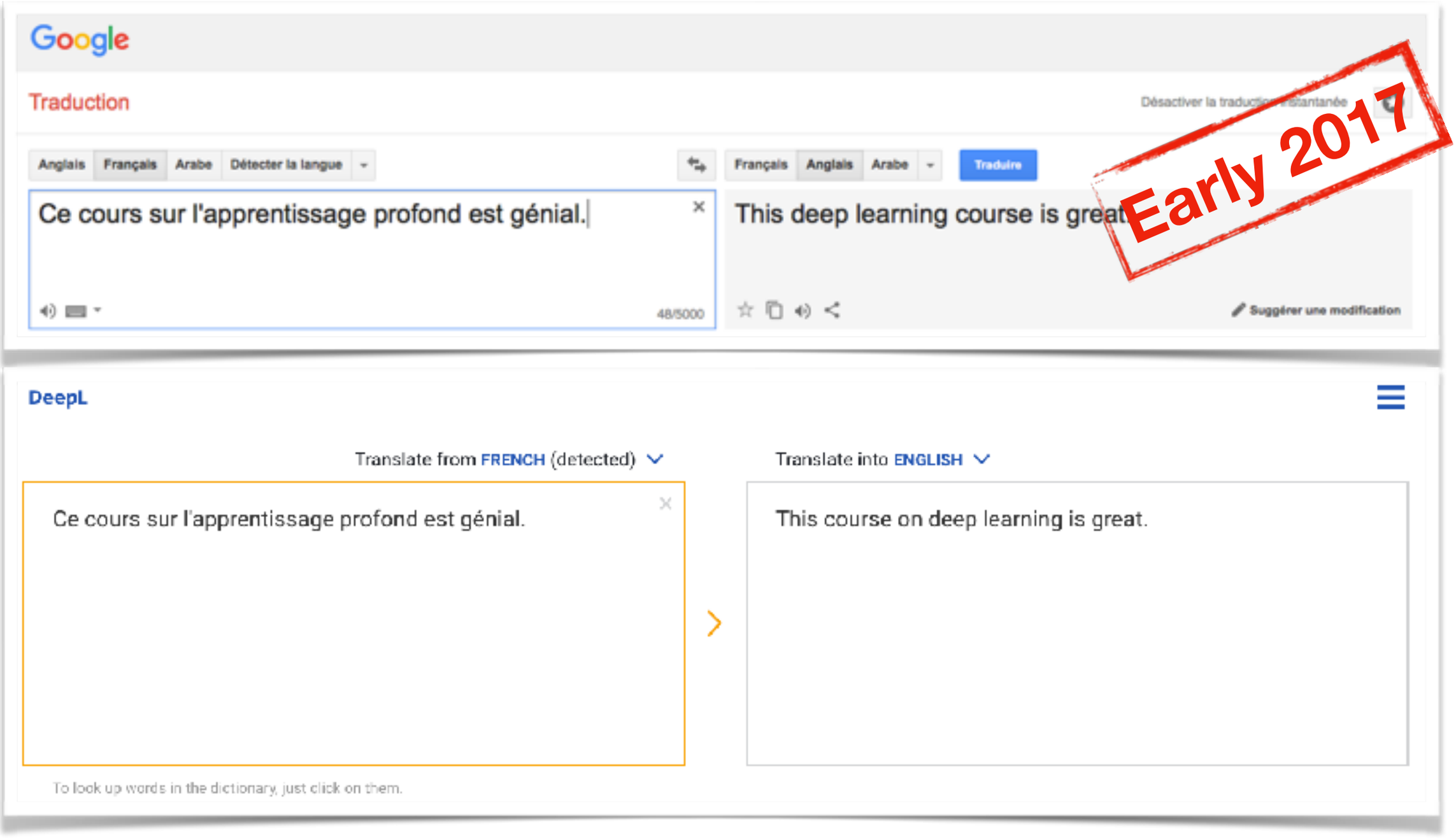
A **dog** is standing on a hardwood floor.

A little **girl** sitting on a bed with a teddy bear.

A group of **people** sitting on a boat in the water.

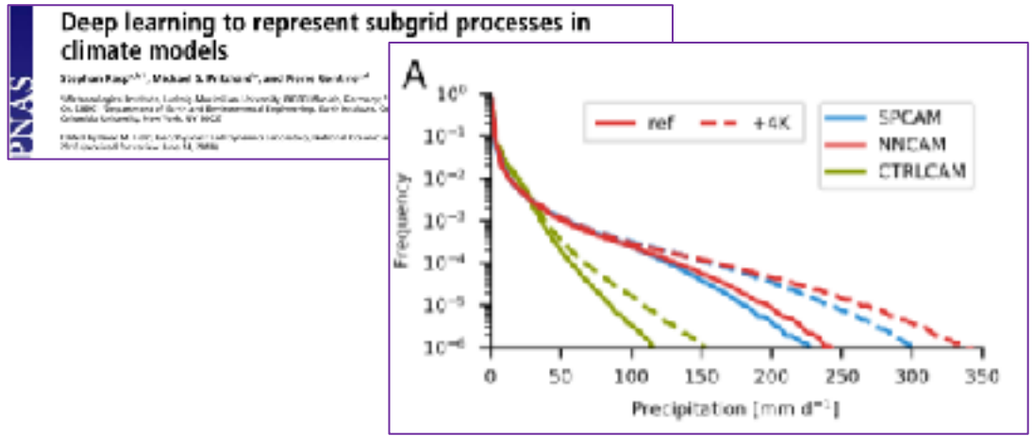
Vinyals et al., ICML 2014

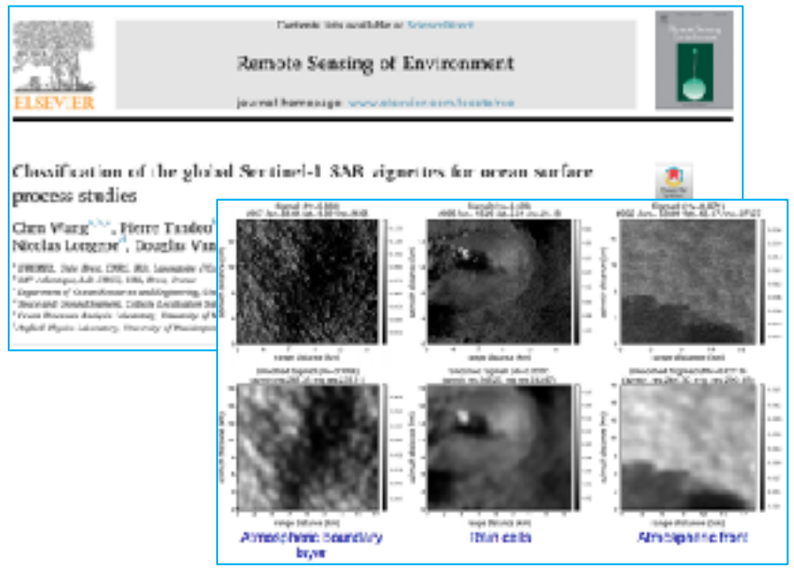
Automatic Translation

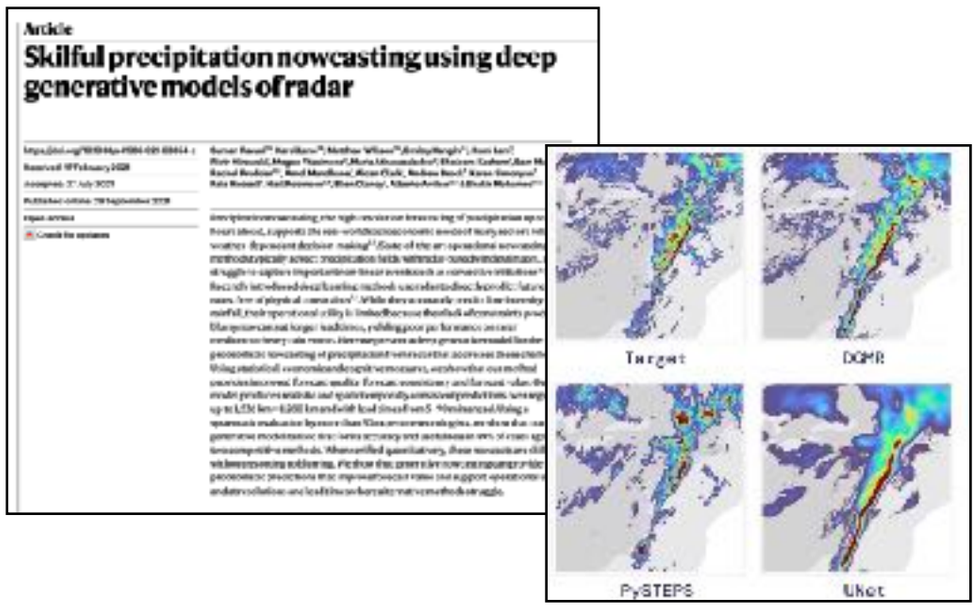


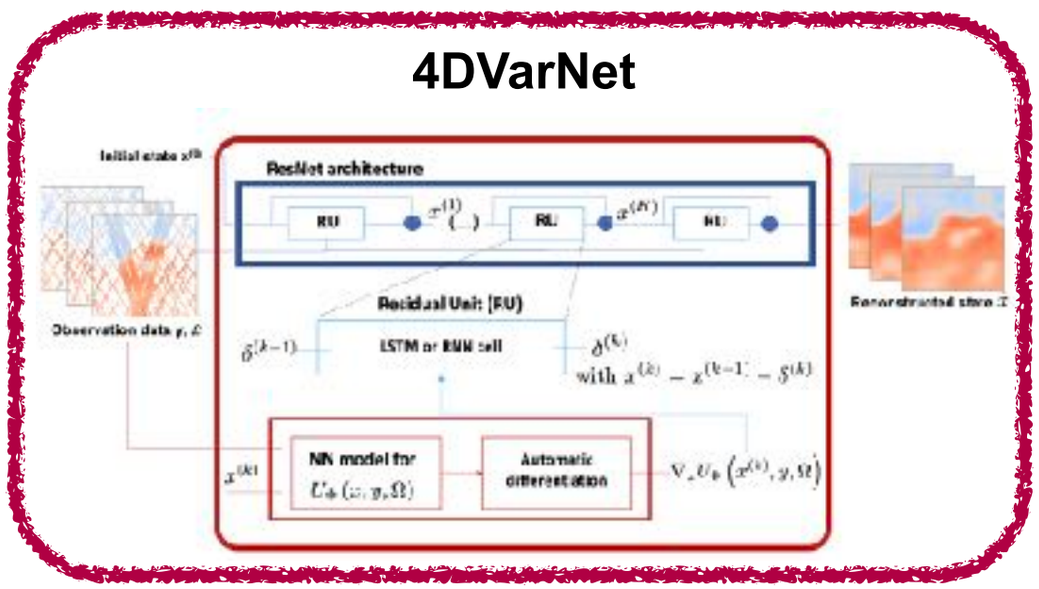
and many more…….

Deep learning in Ocean/ atmosphere Science



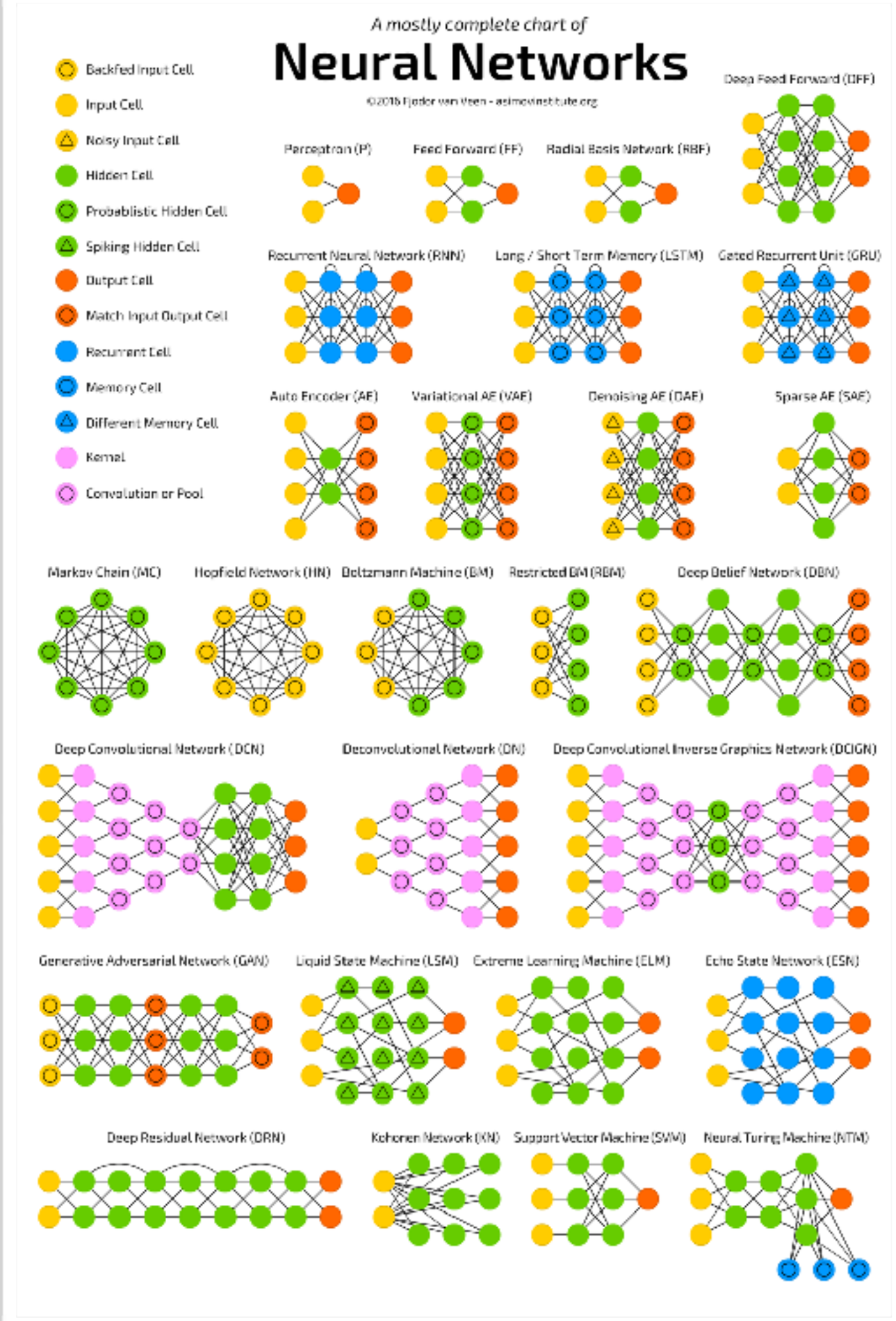




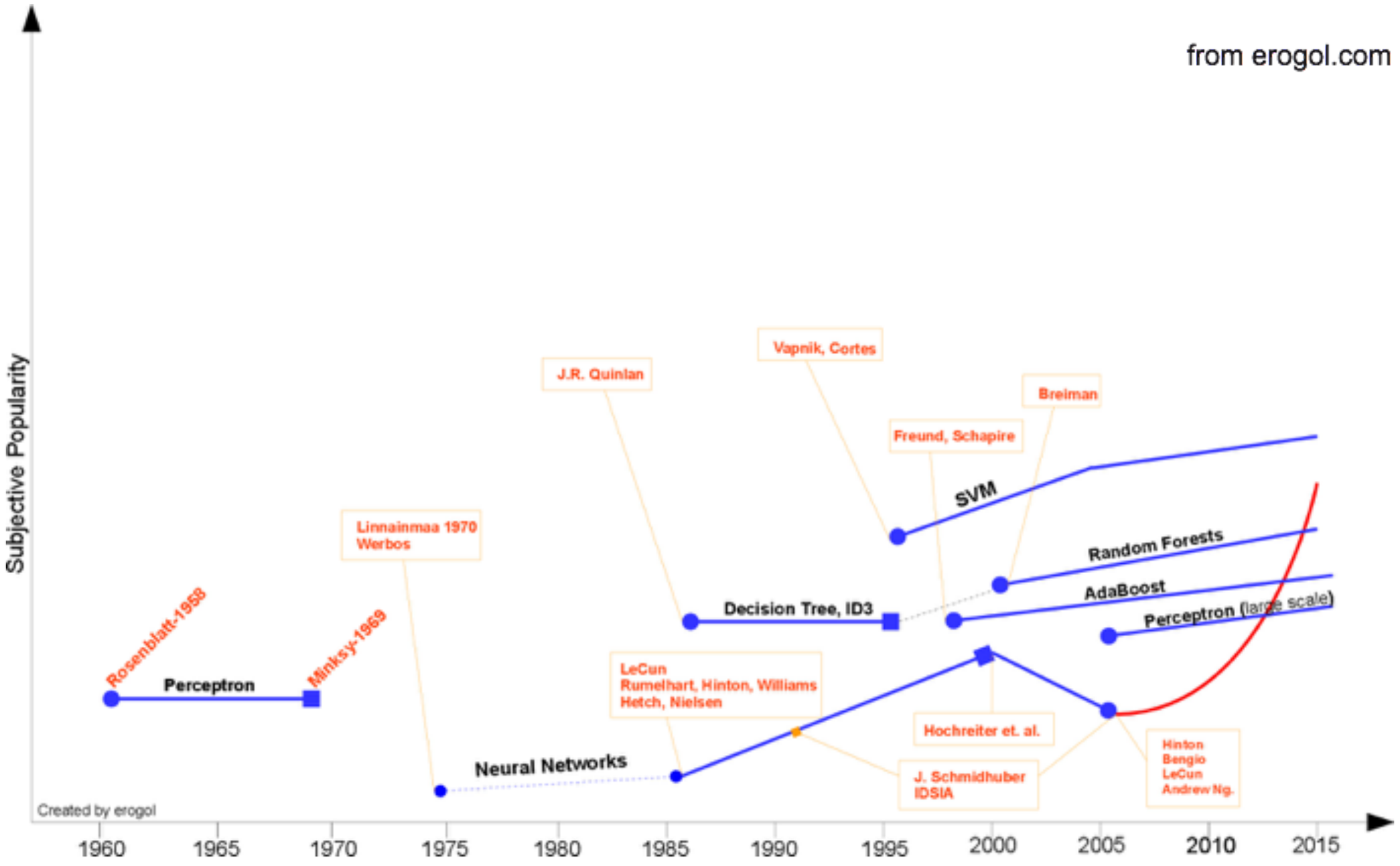


Deep learning relies on neural networks, which are not new…

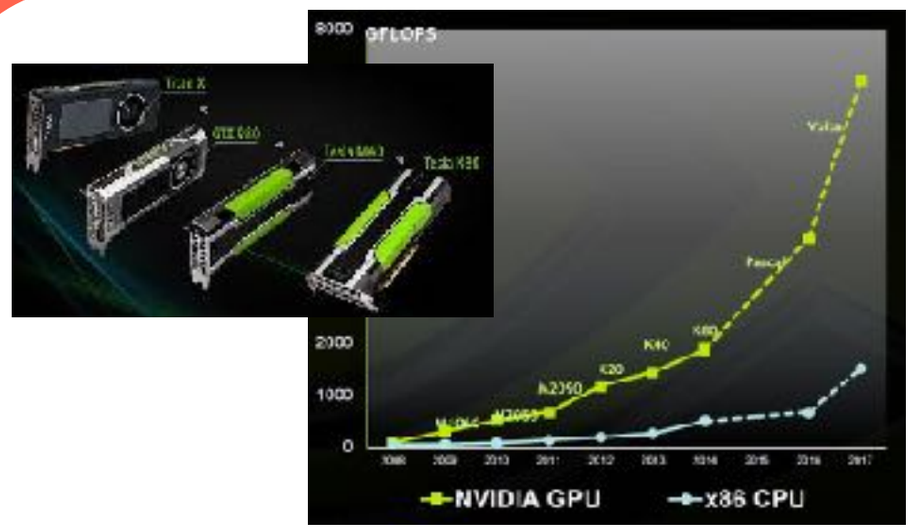
Artificial neural networks date back to the 60's and were popular in the 80's.



Short history



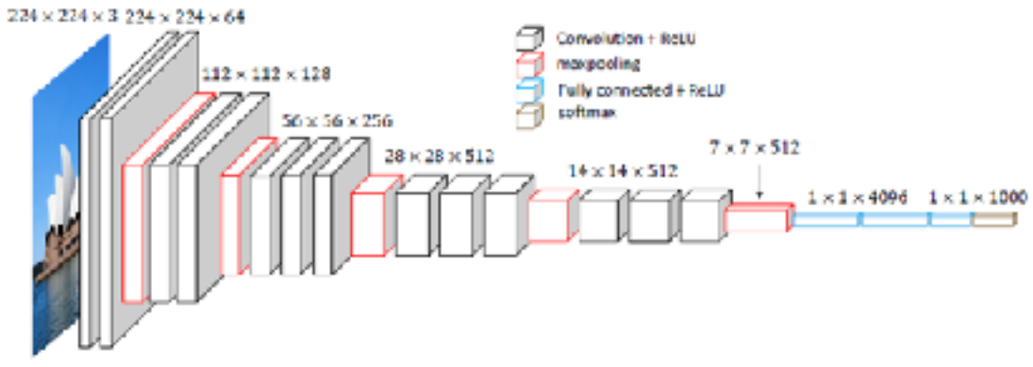
Key reasons for DL emergence



High-performance computing (GPU)

Large annotated dataset (> 1M)

Efficient & easy-to-use libraries



End-to-end learning

History of Deep Learning

Neural Networks 61 (2015) 85–117

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Review

Deep learning in neural networks: An overview

Jürgen Schmidhuber

The Swiss AI Lab IDSIA, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, University of Lugano & SUPSI, Galleria 2, 6928 Manno-Lugano, Switzerland

ARTICLE INFO

Article history:
Received 2 May 2014
Received in revised form 12 September 2014
Accepted 14 September 2014
Available online 13 October 2014

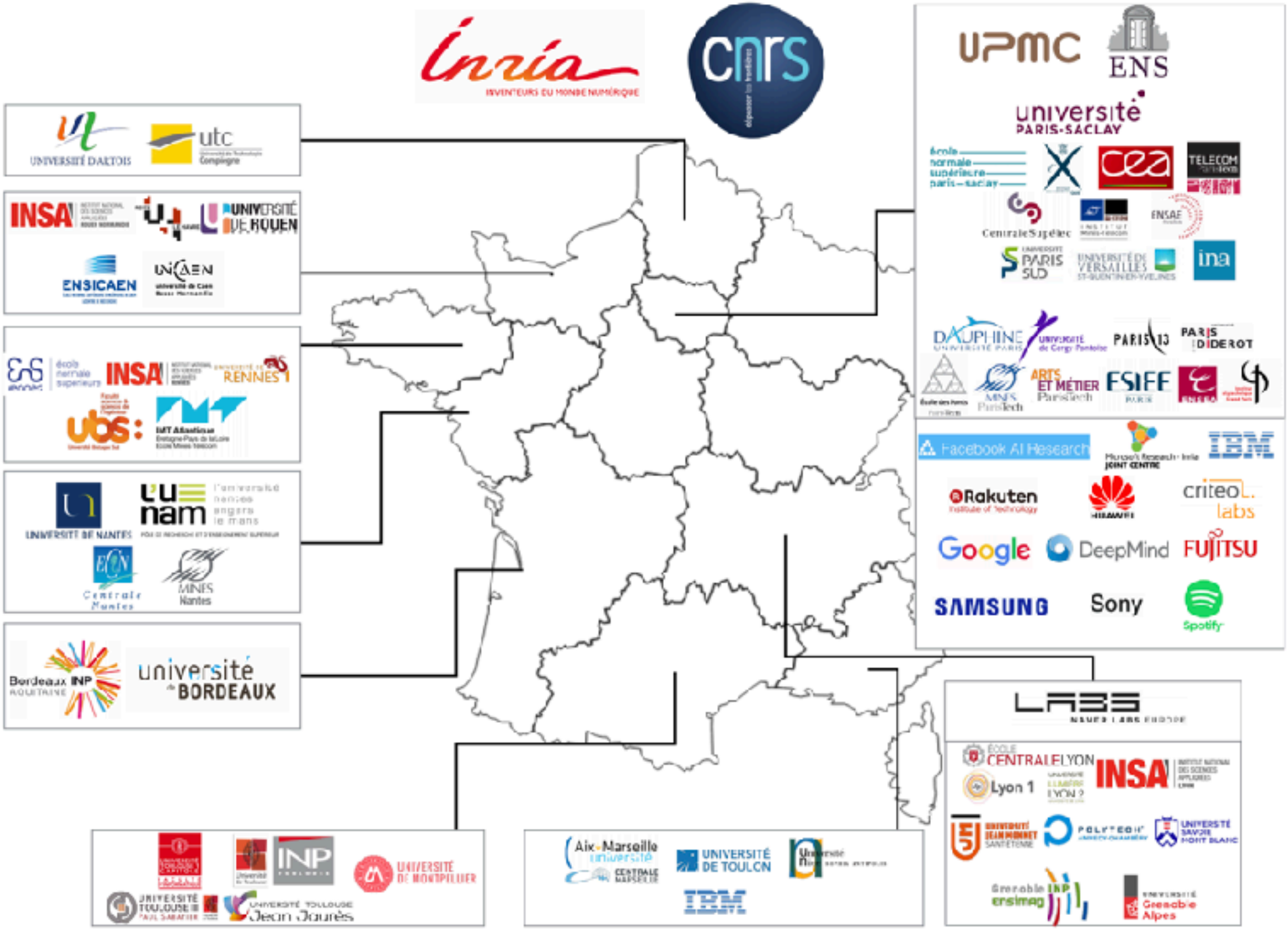
Keywords:
Deep learning
Supervised learning
Unsupervised learning
Reinforcement learning
Evolutionary computation

ABSTRACT

In recent years, deep artificial neural networks (including recurrent ones) have won numerous contests in pattern recognition and machine learning. This historical survey compactly summarizes relevant work, much of it from the previous millennium. Shallow and Deep Learners are distinguished by the depth of their *credit assignment paths*, which are chains of possibly learnable, causal links between actions and effects. I review deep supervised learning (also recapitulating the history of backpropagation), unsupervised learning, reinforcement learning & evolutionary computation, and indirect search for short programs encoding deep and large networks.

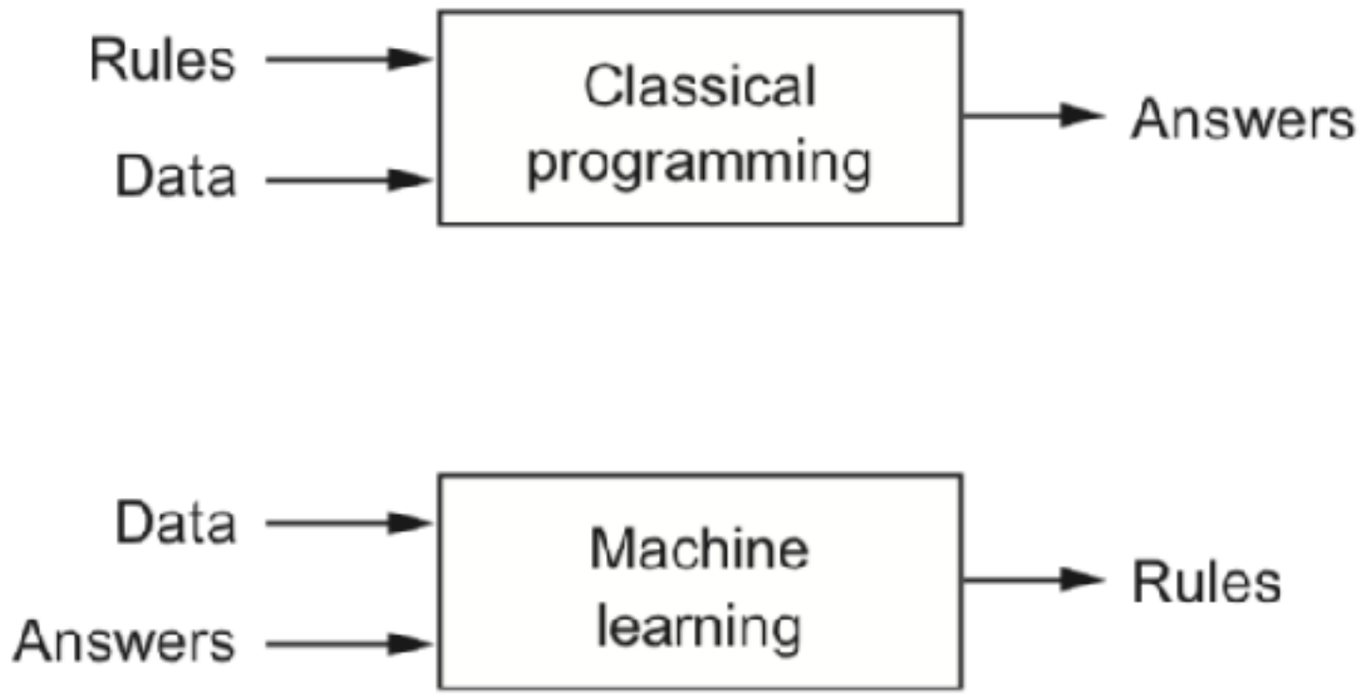
france is AI

AI today
research in France



WHAT DOES (MACHINE) LEARNING MEAN ?

Learning *from* data



[Chollet 2018]

Machine Learning

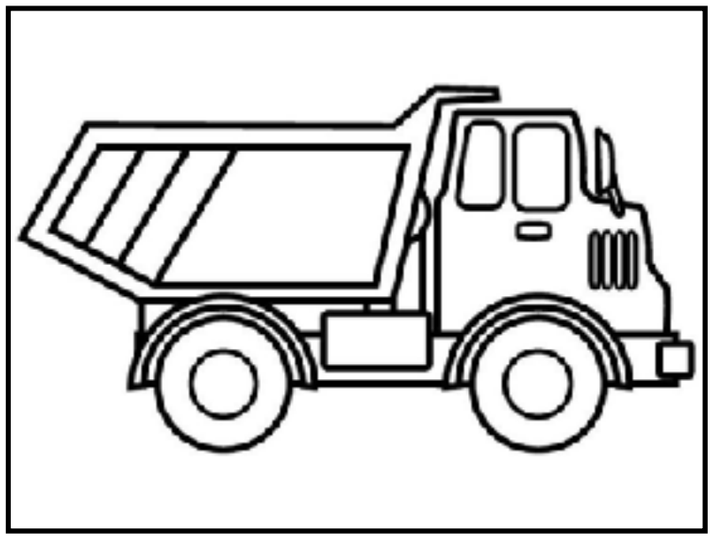
« A computer program is said to learn from experience **E** with respect to some task **T** and some performance measure **P**, if its performance on **T**, as measured by **P**, improves with experience **E**. » (Tom Mitchell, 1998)

Machine Learning Tasks:

- **supervised learning**
- **unsupervised learning**

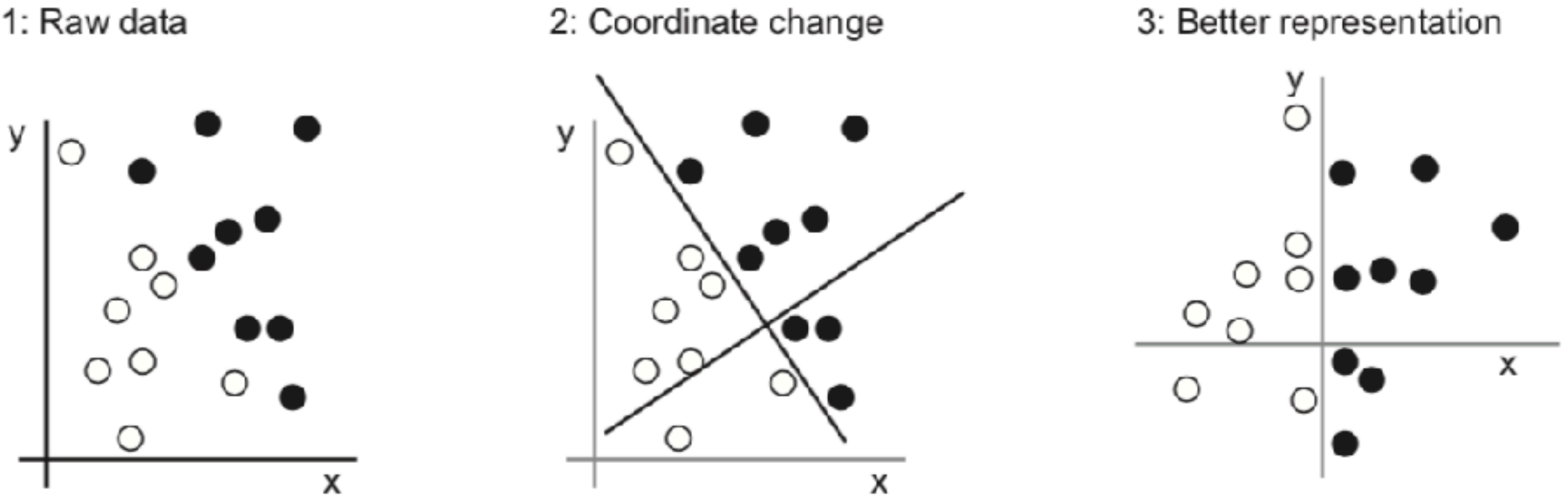
input **output**

CAR



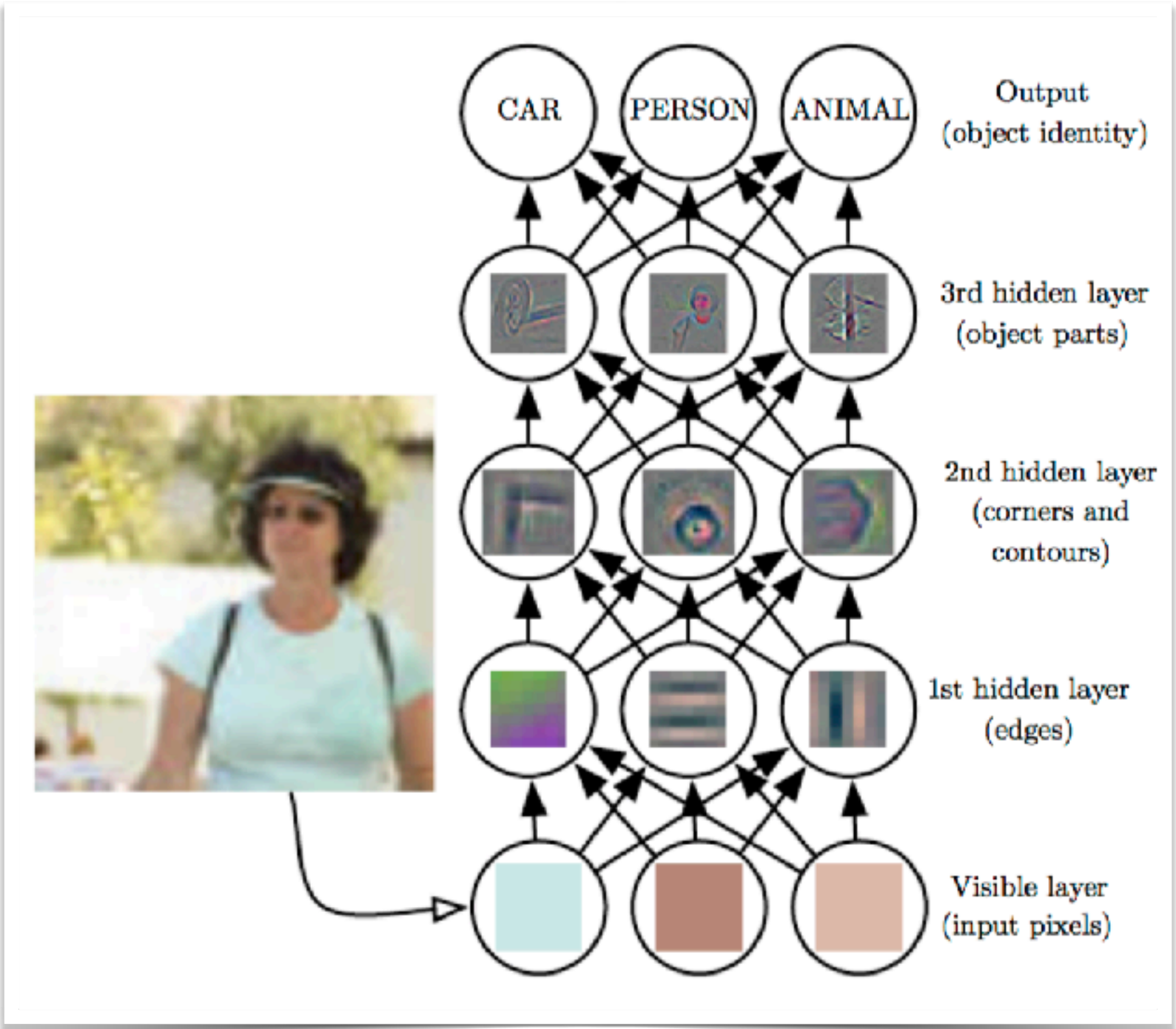
TRUCK

Data representation

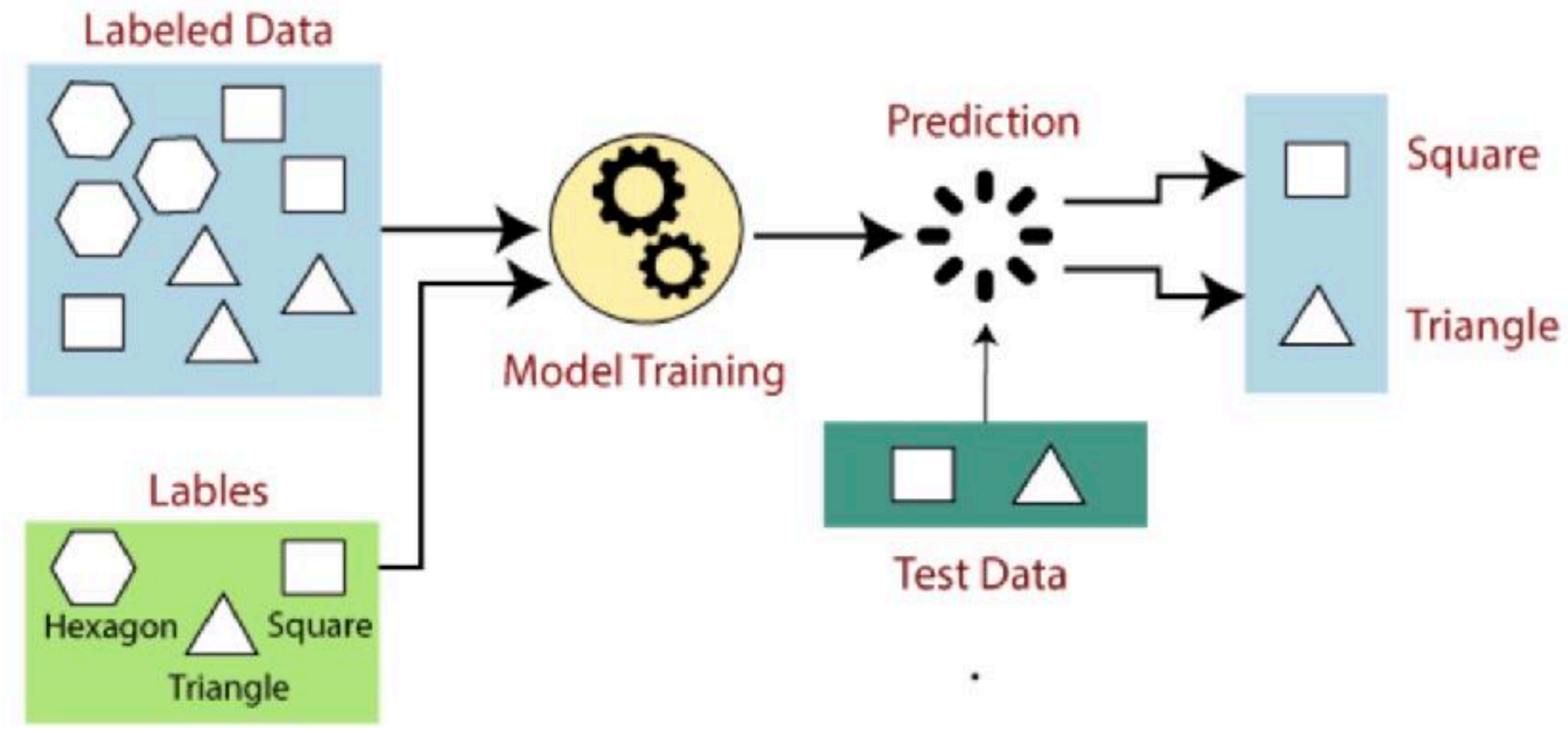


[Chollet 2018]

What is Deep Learning?



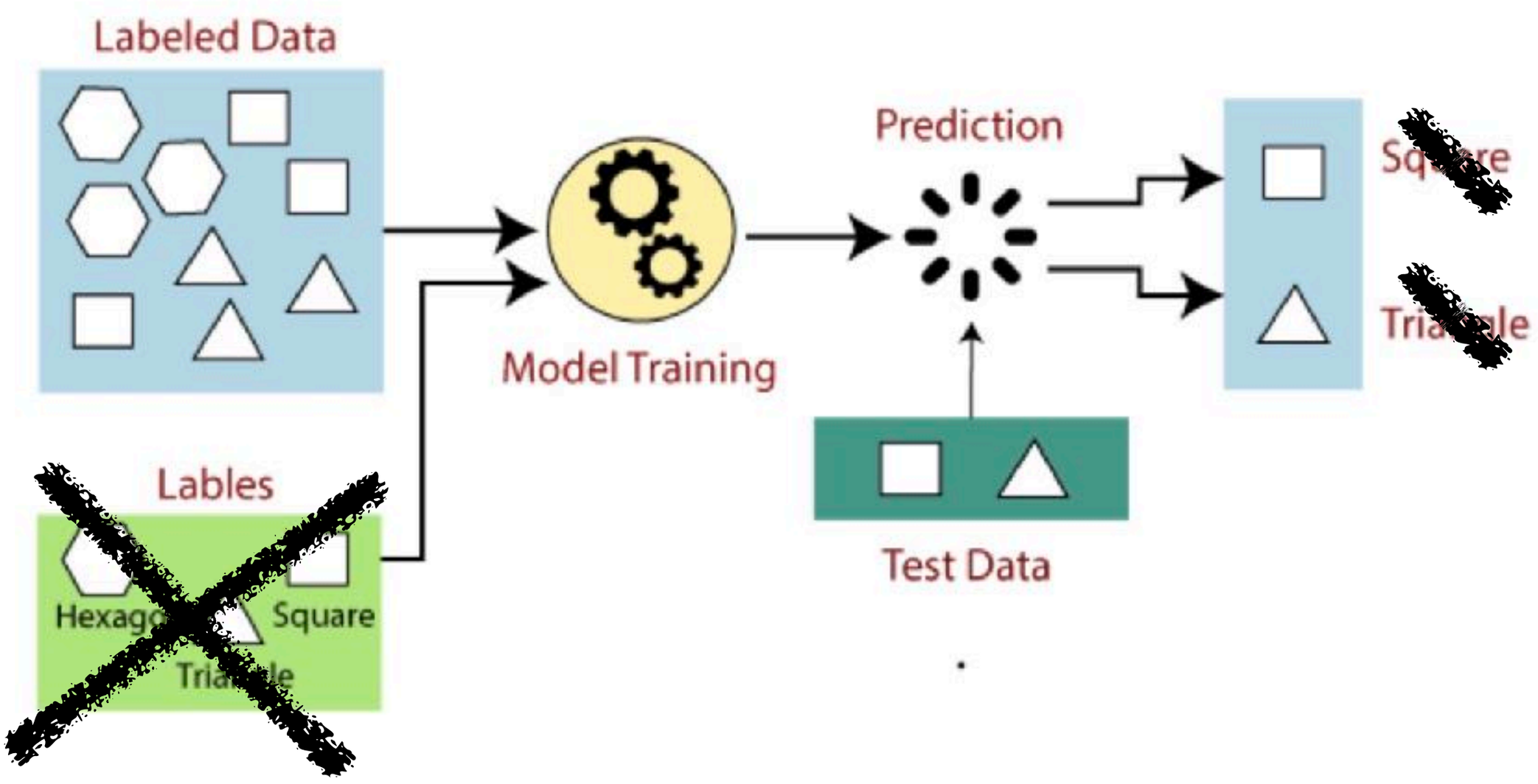
Supervised learning



Within the training dataset, each data is associated with a known label such that the training procedure can evaluate the quality of the current model in terms of a performance score for the prediction of the true labels.

From https://www.javatpoint.com/supervised-machine-learning

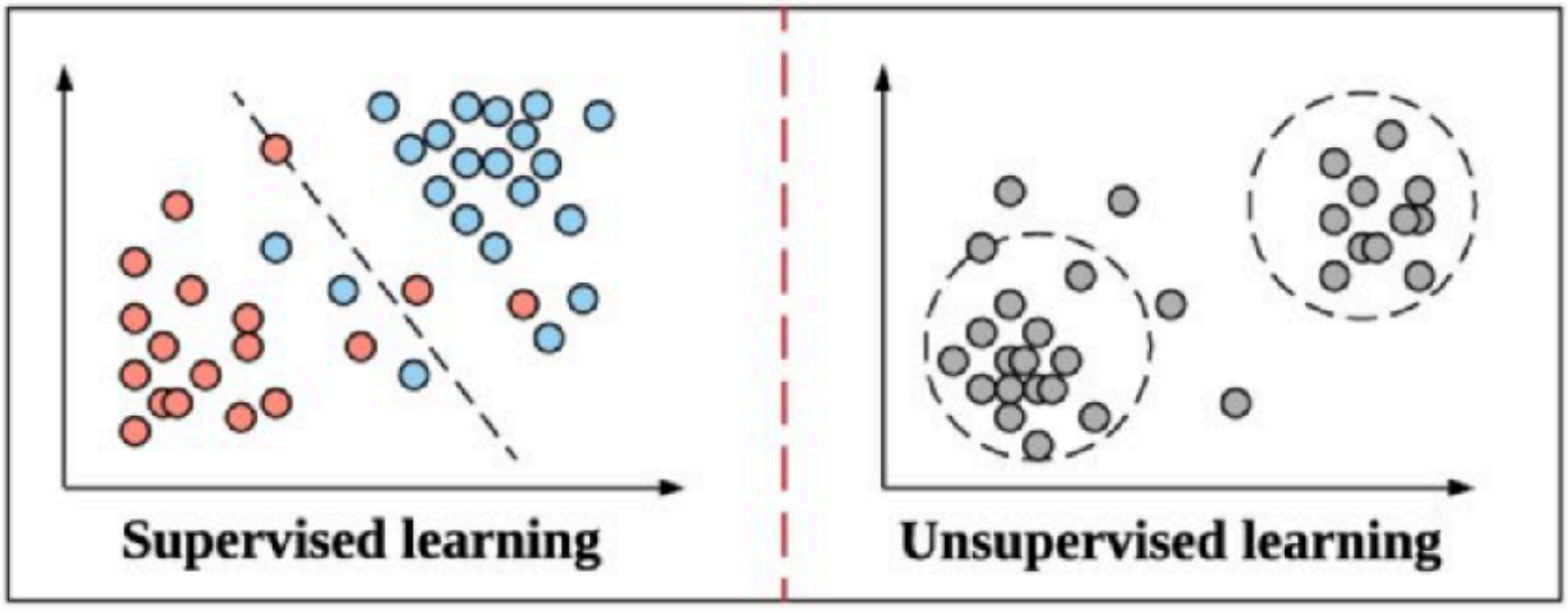
Unsupervised learning



The training dataset does not involve groundtruthed labels.

From https://www.javatpoint.com/supervised-machine-learning

Supervised vs. Unsupervised learning

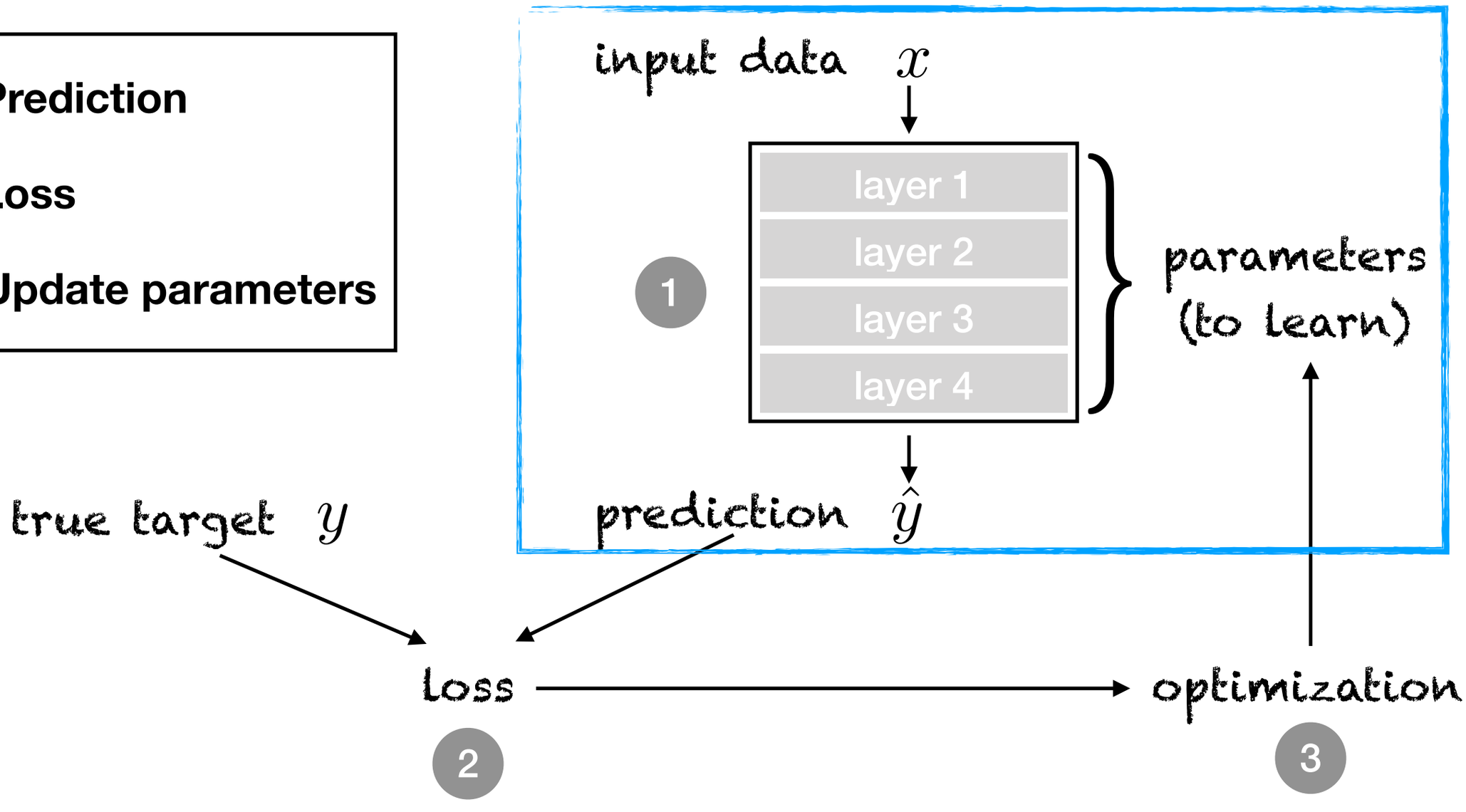


Qian et al., 2019 Link

WHAT DOES (MACHINE) LEARNING MEAN FROM A MATHEMATICAL POINT OF VIEW ?

Machine learning

1. **Prediction**
2. **Loss**
3. **Update parameters**



Machine/Deep Learning

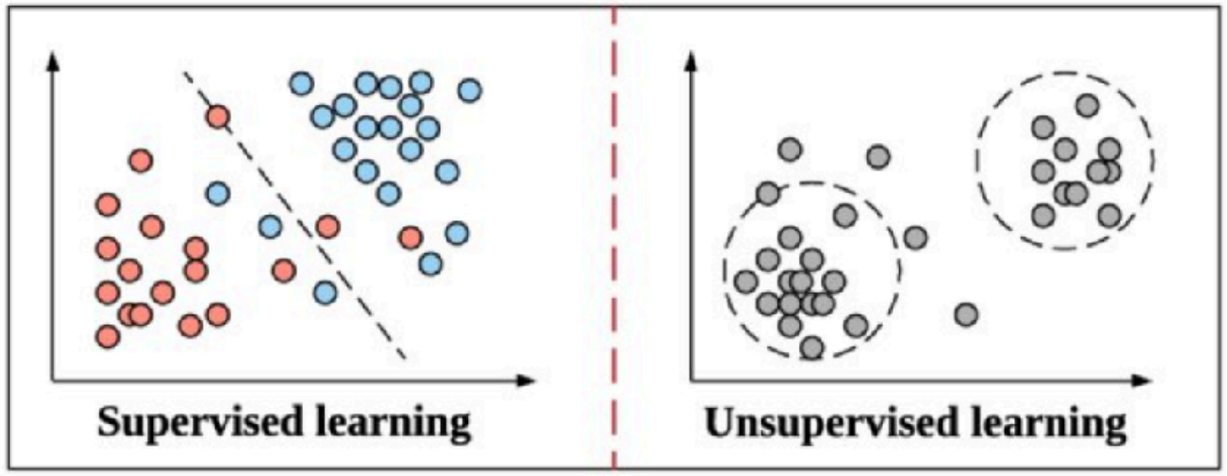
Mathematical formulation: Learning comes to minimising some loss function given w.r.t. model parameters and training data

$$\widehat{\theta} = \arg\min_{\theta} \mathcal{L}\left(\{x_i, y_i\}_{i \in \{1,..,N\}}; f_\theta\right)$$

Key questions:

- Which parameterisation for model f ?
- Which loss function ?

Supervised vs. Unsupervised learning



Which training loss ?

Qian et al., 2019 Link

Machine Learning Terminology

- Supervised learning (*Apprentissage supervisé*)
- Unsupervised learning (*Apprentissage non-supervisé*)
- Training and test dataset (*Jeux de données d'apprentissage et test*)
- Training loss (*Coût d'apprentissage*)
- Model (*Modèle*)
- Regression
- Classification

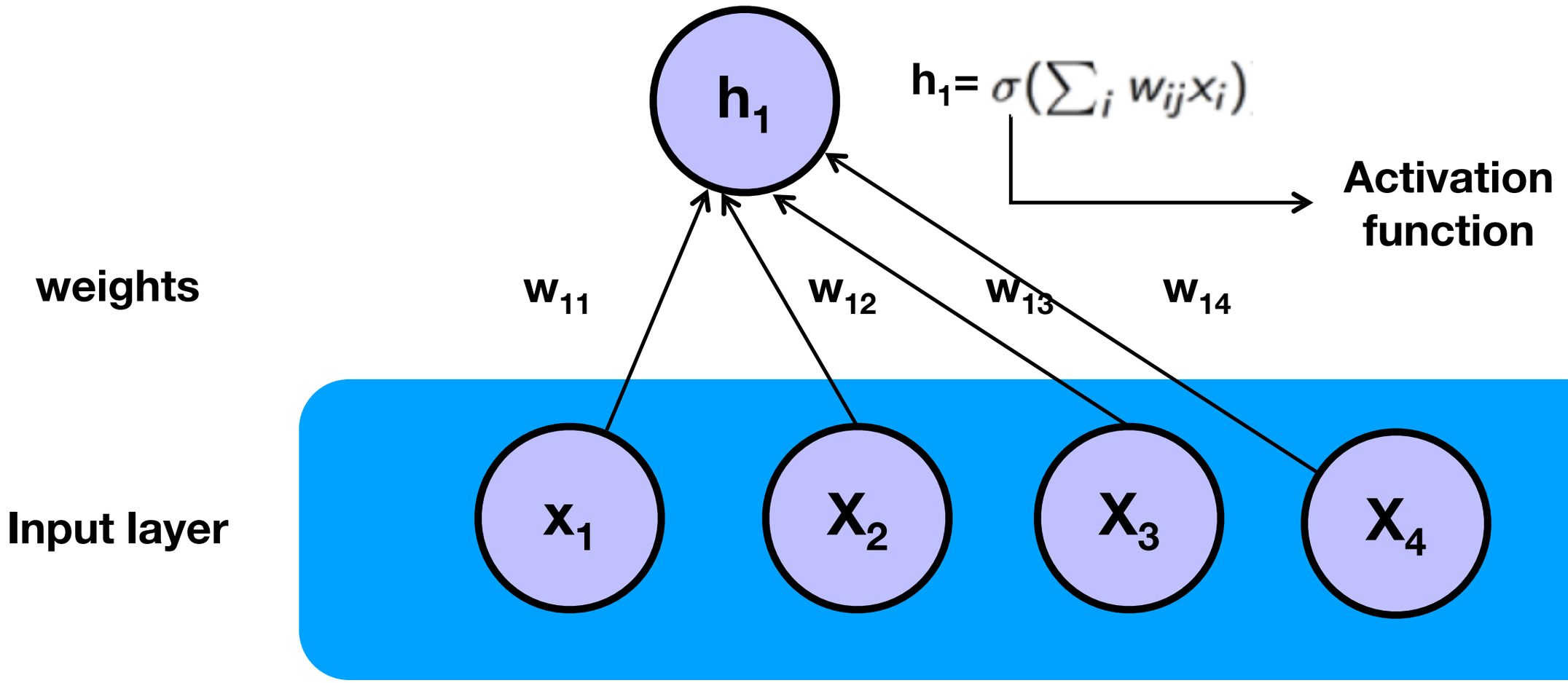
NEURAL NETWORKS:
KEY PRINCIPLES

Feedforward networks

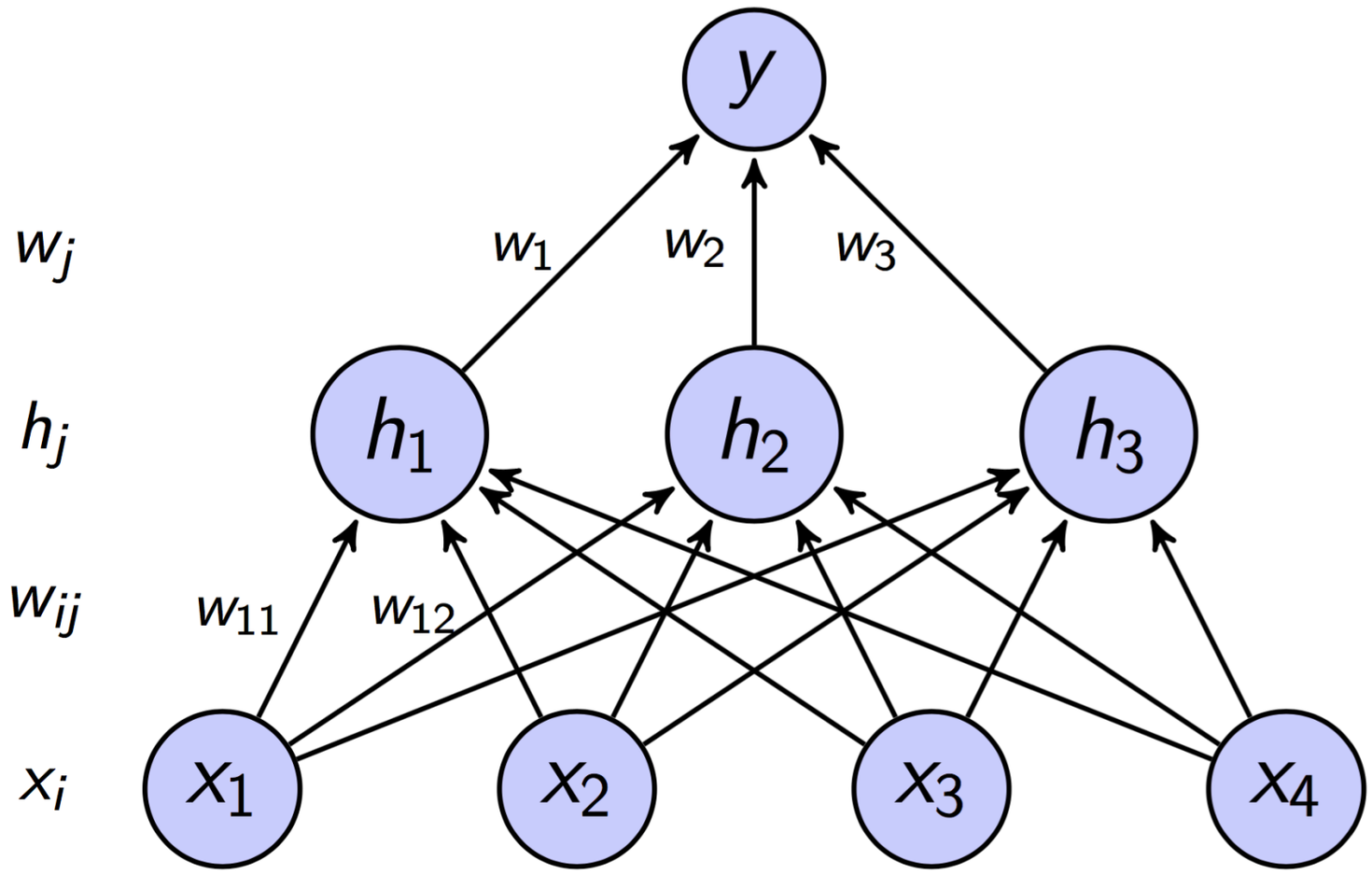
- Objective: approximate a function f^*
- How: learning a mapping: $\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$

Parameters to learn

The artificial neuron

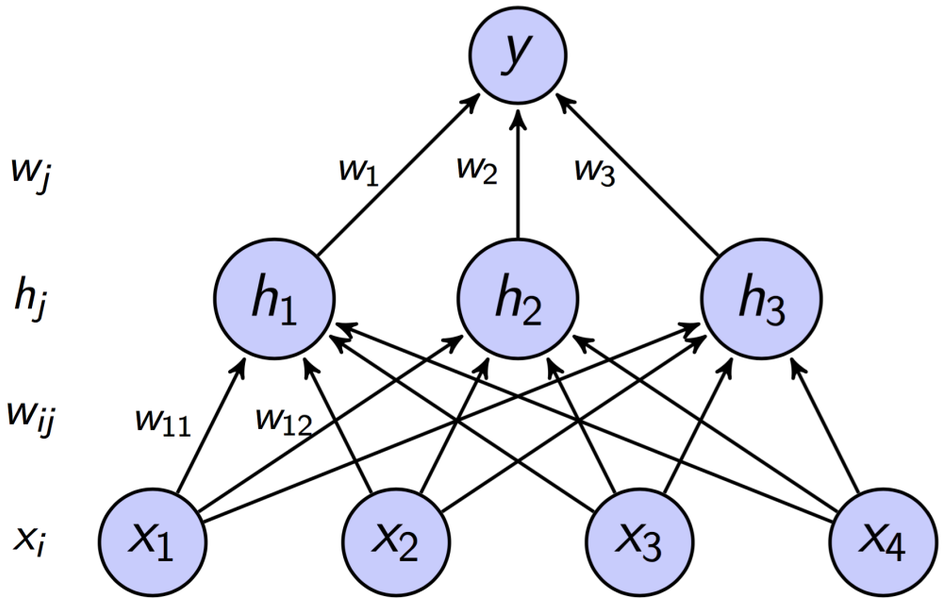


Feedforward networks



$$f(x) = \sigma(\sum_j w_j \cdot h_j) = \sigma(\sum_j w_j \cdot \sigma(\sum_i w_{ij} x_i))$$

In practice: network architecture



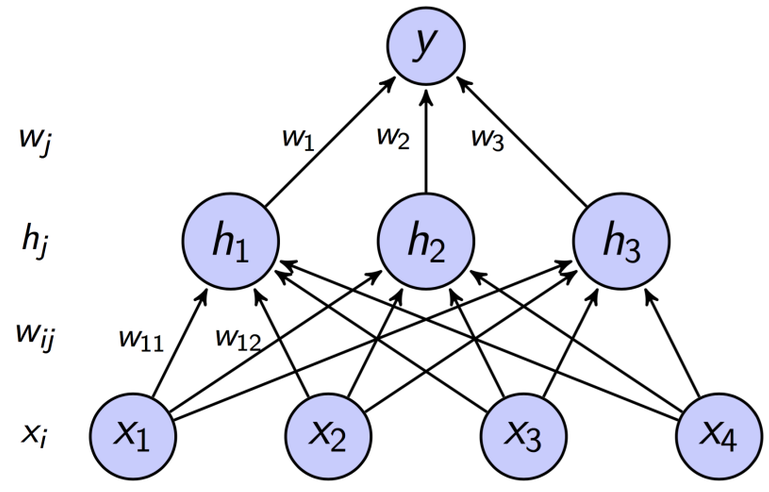

```
from keras import models
from keras import layers
```

```
network = models.Sequential()
network.add(layers.Dense(3, activation='relu', input_shape=(4,)))
network.add(layers.Dense(1, activation='softmax'))
```

Feedforward networks

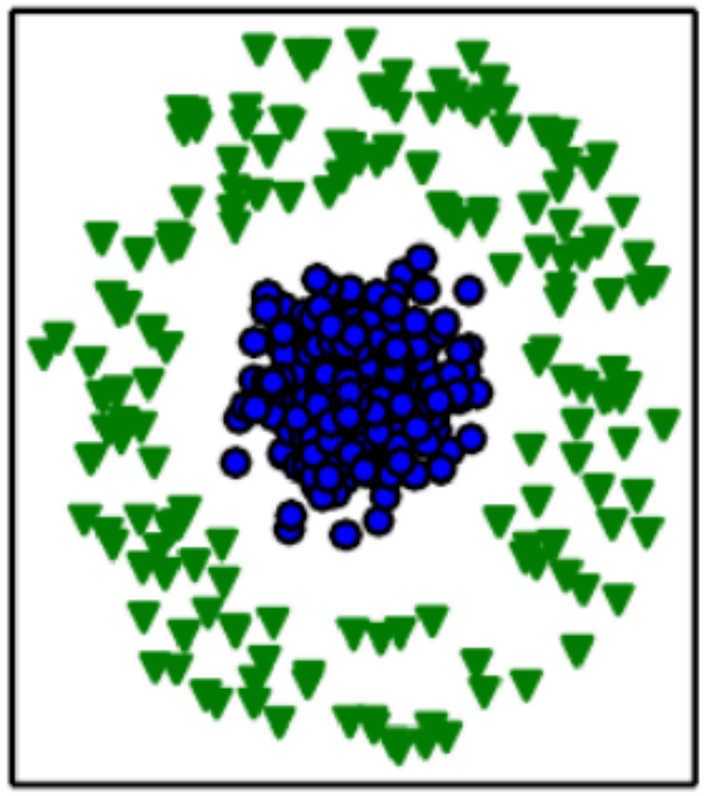
- Feedforward: no feedback connection

- Network: composition of functions



- Neuron: basic unit, inspired from neuroscience

Feedforward networks



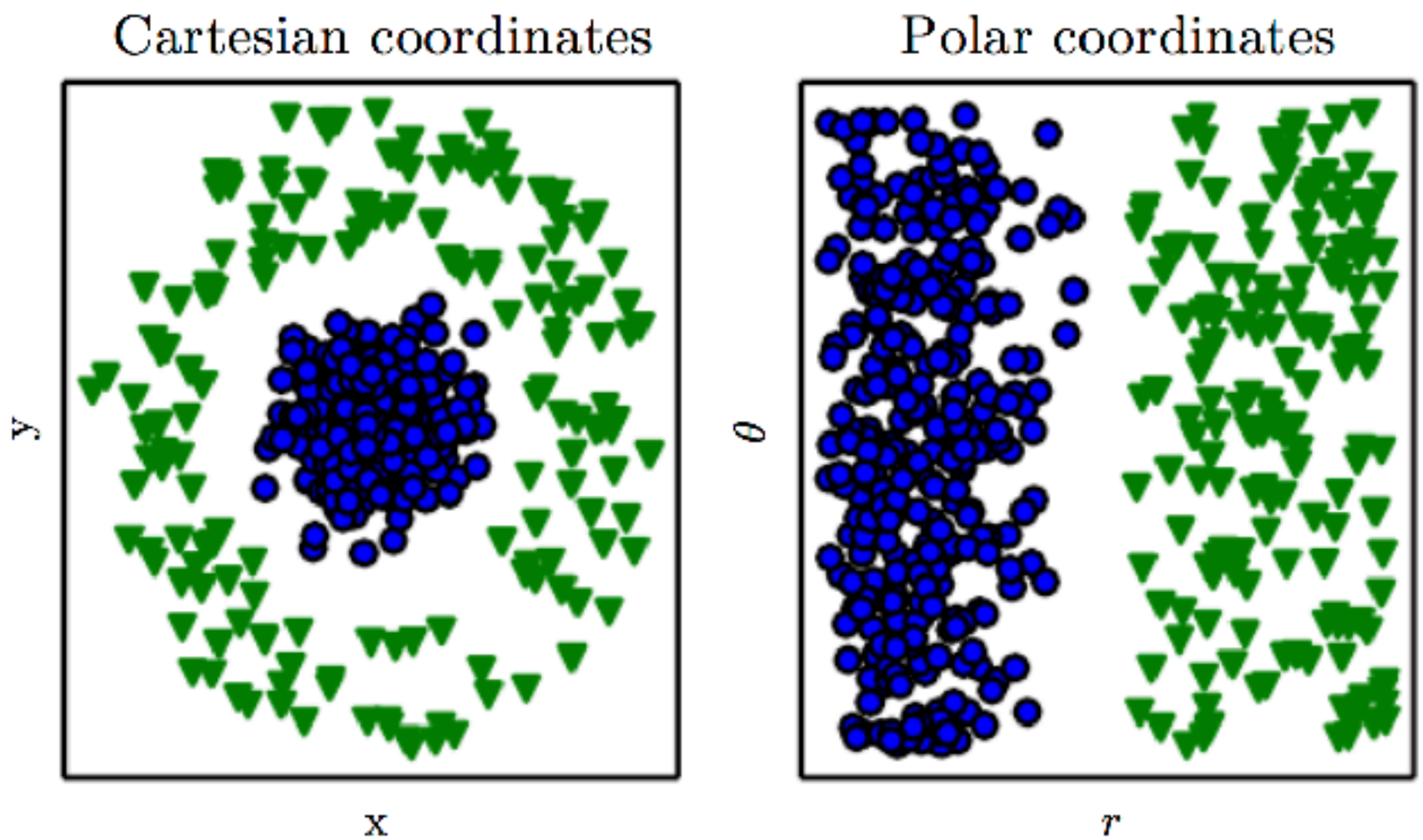
[Goodfellow et al. 2016]

Feedforward networks

- Linear models: $\boldsymbol{y} = \boldsymbol{w}^\top \boldsymbol{x}$
  - efficient, reliable
  - closed form or convex optimization
- Non linear models in deep learning:

$$\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{w}) = \phi(\boldsymbol{x}; \boldsymbol{\theta})^\top \boldsymbol{w}$$

Feedforward networks

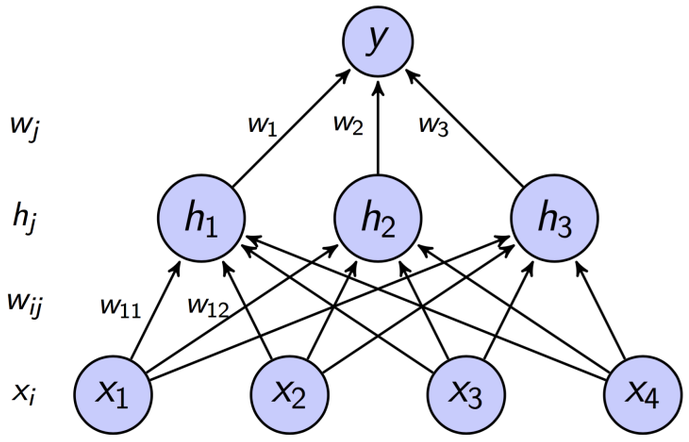


$$\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{w}) = \phi(\boldsymbol{x}; \boldsymbol{\theta})^\top \boldsymbol{w}$$

[Goodfellow et al. 2016]

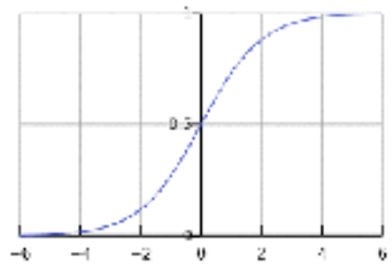
Architecture design

Activation functions



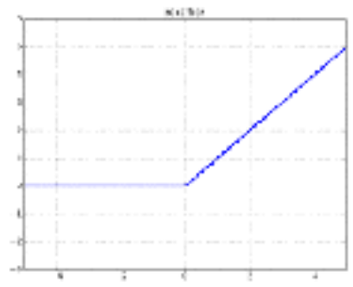
$$f(x) = \sigma(\sum_j w_j \cdot h_j) = \sigma(\sum_j w_j \cdot \sigma(\sum_i w_{ij} x_i))$$

softmax



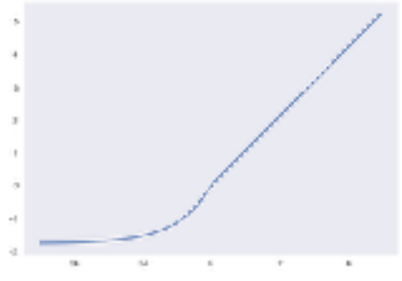
$$\sigma : \mathbb{R}^K \to (0,1)^K$$
$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

ReLU



$$f(x) = x^+ = \max(0, x)$$

selu



$$\text{selu}(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha e^x - \alpha & \text{if } x \leqslant 0 \end{cases}$$

etc.

A mostly complete chart of

Neural Networks

- Backfed Input Cell
- Input Cell
- Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Different Memory Cell
- Kernel
- Convolution or Pool

Perceptron (P)

Feed Forward (FF)

Radial Basis Network (RBF)

Deep Feed Forward (DFF)

Recurrent Neural Network (RNN)

Long / Short Term Memory (LSTM)

Gated Recurrent Unit (GRU)

Auto Encoder (AE)

Variational AE (VAE)

Denoising AE (DAE)

Sparse AE (SAE)

Markov Chain (MC)

Hopfield Network (HN)

Boltzmann Machine (BM)

Restricted BM (RBM)

Deep Belief Network (DBN)

Deep Convolutional Network (DCN)

Deconvolutional Network (DN)

Deep Convolutional Inverse Graphics Network (DCIGN)

Generative Adversarial Network (GAN)

Liquid State Machine (LSM)

Extreme Learning Machine (ELM)

Echo State Network (ESN)

Deep Residual Network (DRN)

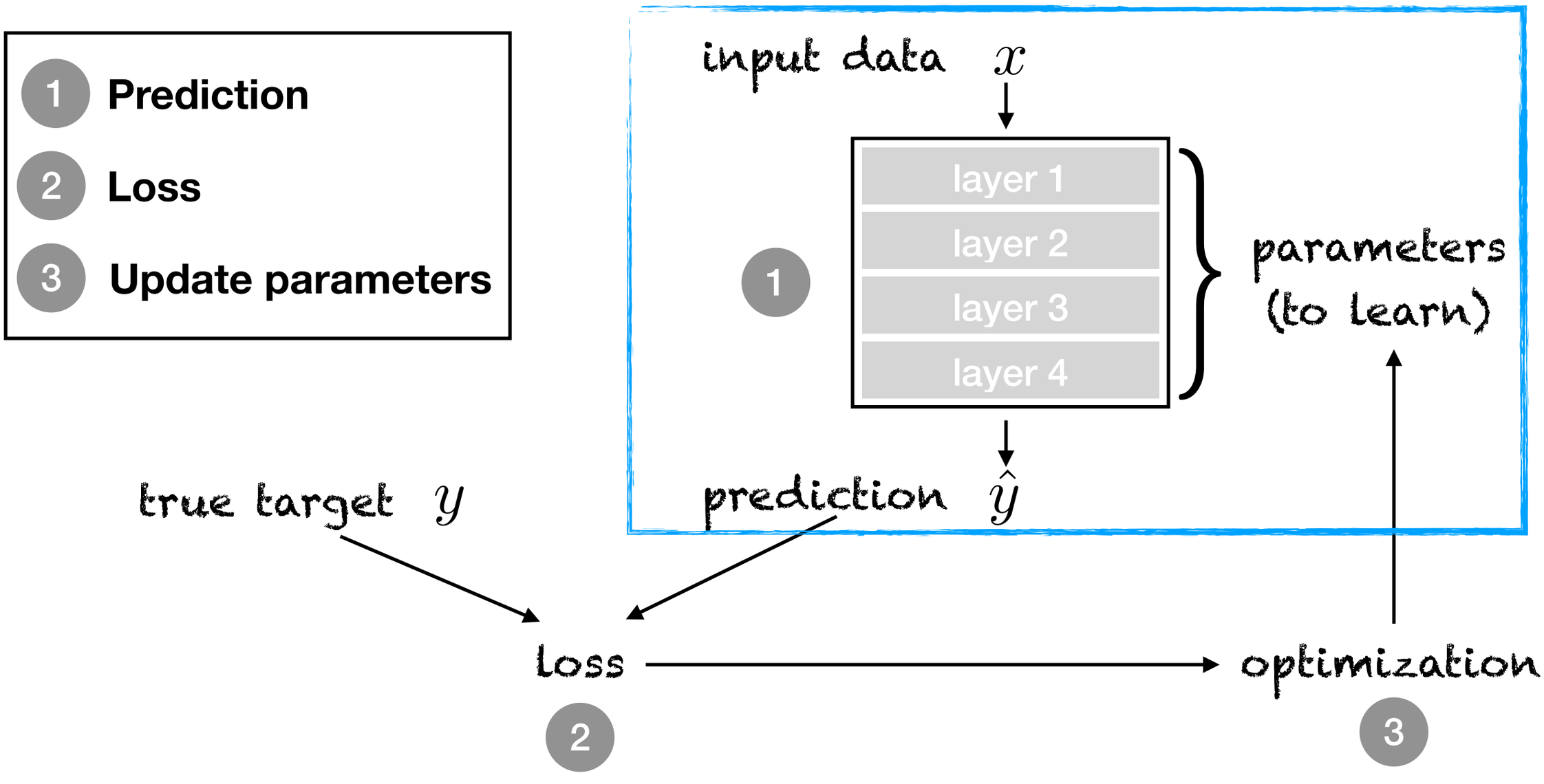
Kohonen Network (KN)

Support Vector Machine (SVM)

Neural Turing Machine (NTM)

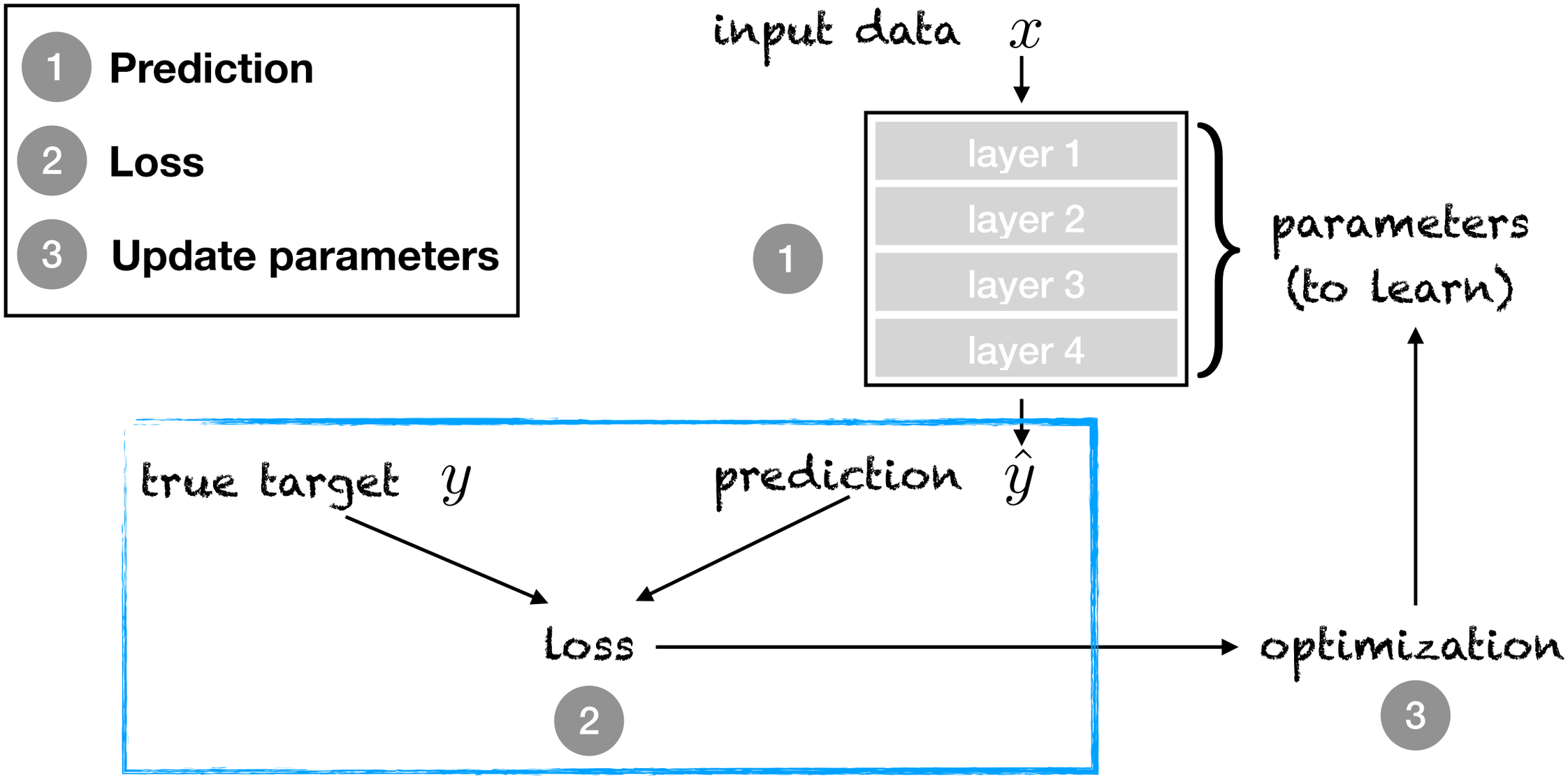
Overview

1. **Prediction**
2. **Loss**
3. **Update parameters**



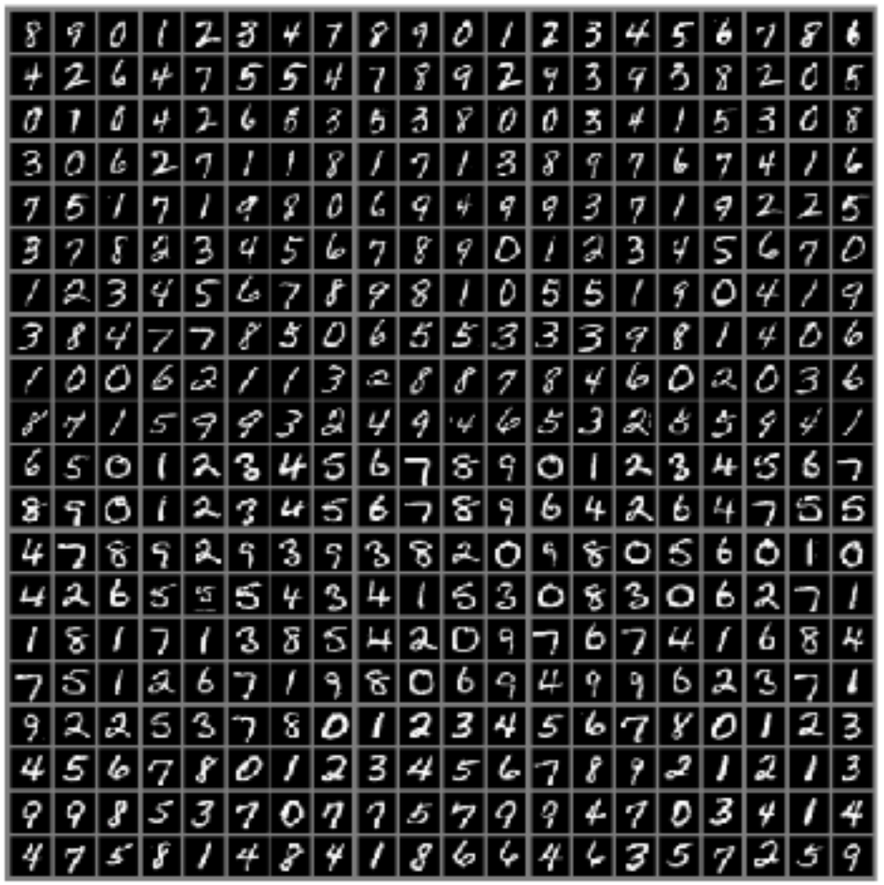
Overview

1. **Prediction**
2. **Loss**
3. **Update parameters**



Cost function

MNIST dataset
  input: image
  output: a class (one-hot encoding)



How to choose the loss function ?

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} |y^{(i)} - f(x^{(i)};\theta)|$$

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \left(y^{(i)} - f(x^{(i)};\theta)\right)^2$$

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log f(x^{(i)};\theta)$$

etc.

Cost Function

(also called loss function or objective function)

- Empirical risk (training loss)

$$\mathbb{E}_{(x,y)\sim\hat{p}_{data}} L(f(x;\theta),y) = \frac{1}{m}\sum_{i=1}^{m} L(f(x^{(i)};\theta),y^{(i)})$$

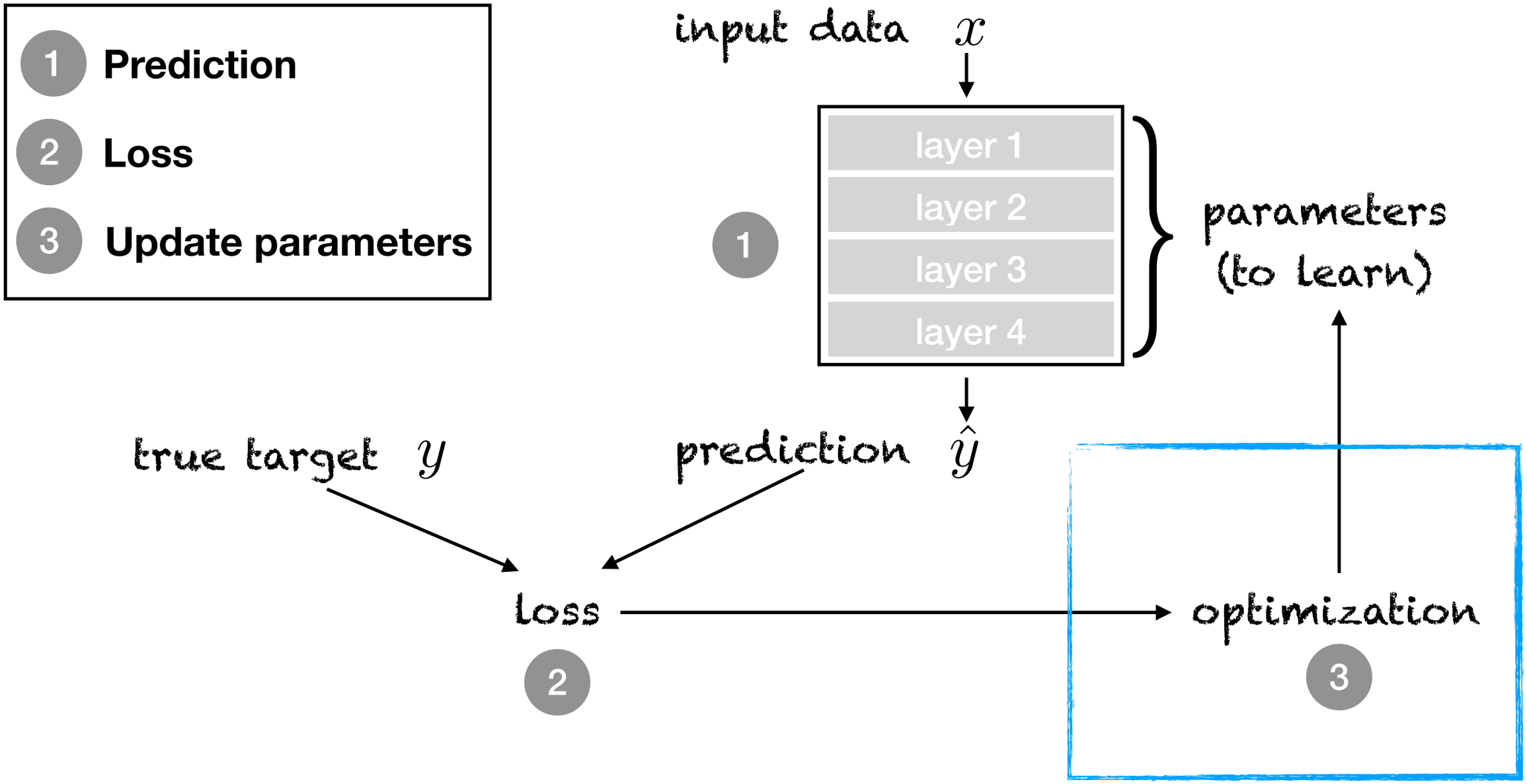
- Learning: We want to minimize J in the hope that doing so will improve the performance measure P.

Let’s play with MLPs using tensorflow playground

http://playground.tensorflow.org/

Overview

1. **Prediction**
2. **Loss**
3. **Update parameters**



Neural networks: composition idea

- **Approximation through the composition of (simple) elementary functions:**

$$f_\theta(x) = f_{\theta_N} \circ \ldots \circ f_{\theta_2} \circ f_{\theta_1}(x)$$

- **Key features:**
  - Any continuous function can be approximated as the composition of elementary functions
  - Analytical/exact computation of the derivative of f with respect to parameters and input variables
  - Direct exploitation of gradient-based optimisation schemes

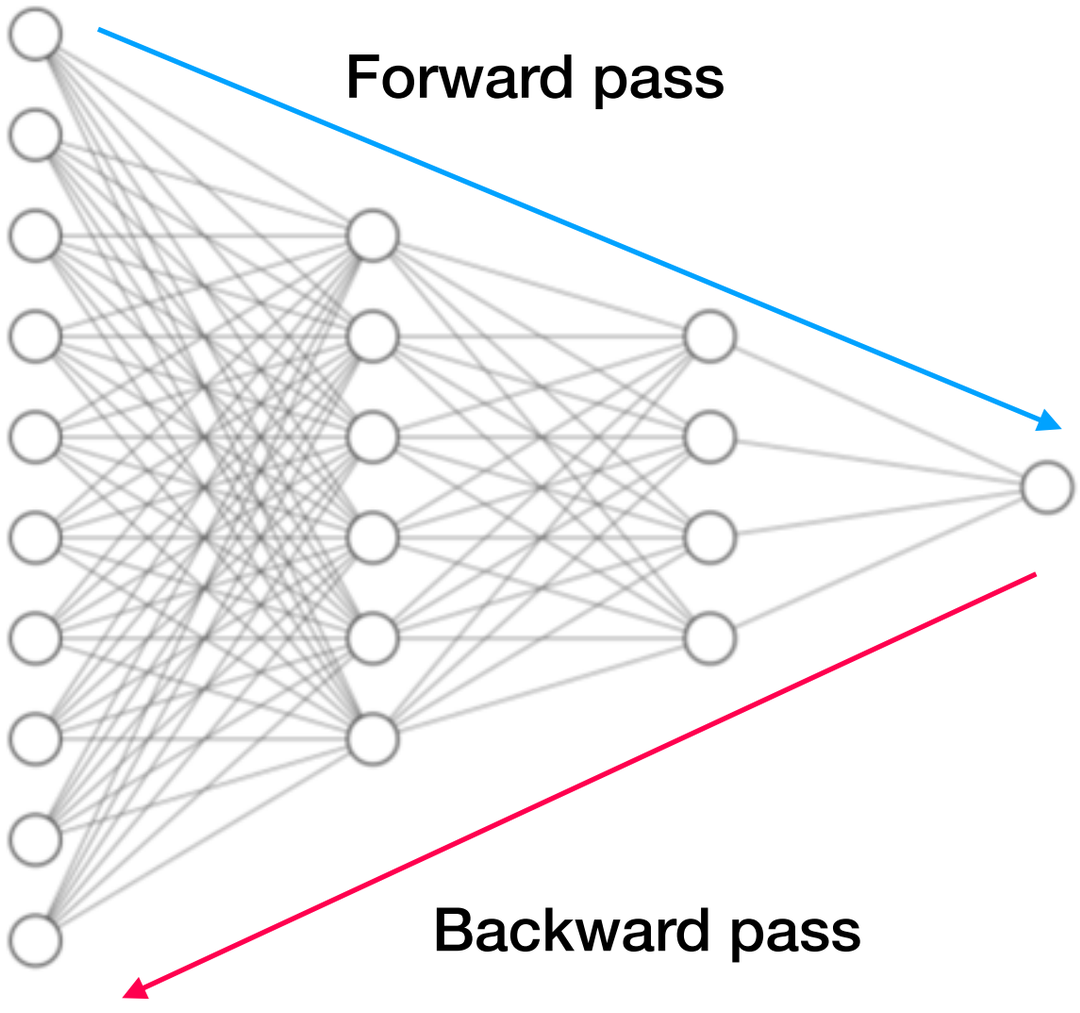
Neural networks: Derivatives, Chain rule and Backpropagation

$$f_\theta(x) = f_{\theta_N} \circ \ldots \circ f_{\theta_2} \circ f_{\theta_1}(x)$$

Recursion for the computation of the derivatives

$$\frac{\partial f_\theta(x)}{\partial \theta_i} = f'_{\theta_N}\left(f_{\overline{\theta}_{N-1}}(x)\right) \frac{\partial f_{\overline{\theta}_{N-1}}(x)}{\partial \theta_i}$$

$$\frac{\partial f_{\overline{\theta}_k}(x)}{\partial \theta_i} = f'_{\theta_k}\left(f_{\overline{\theta}_{k-1}}(x)\right) \frac{\partial f_{\overline{\theta}_{k-1}}(x)}{\partial \theta_i}$$



Neural networks and differentiable programming

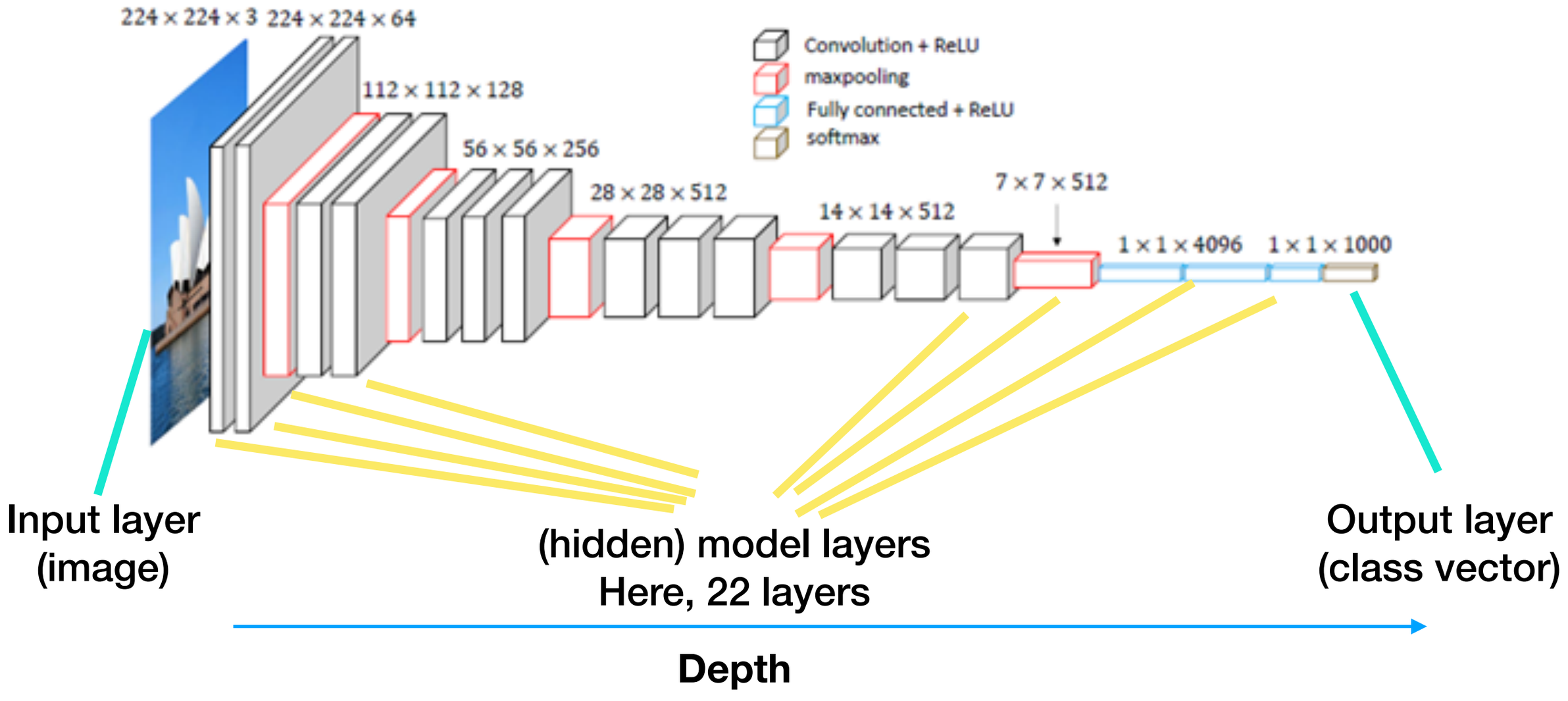
$$f_\theta(x) = f_{\theta_N} \circ \ldots \circ f_{\theta_2} \circ f_{\theta_1}(x)$$

- **Deep learning framework = implementation of library of forward operators (called layers) associated with a backward operator (gradient/adjoint)**

What are deep learning models ?

Basics of DL models

DL models are (in general) feedforward models. VGG16 as an illustration



The more layers, the deeper…… Some models may have up to several hundreds to thousands of layers.

Let’s move to practice

Introduction to PyTorch

Basic tips for pytorch

High-level python library (platform-independent, CPU/GPU, distributed computing,…)

Very similar to numpy syntax but different (eg, torch.tensor vs. numpy.array)

Specific types and classes to implement forward and backward passes through a network

Object-based coding through classes for neural models

Among the two main state-of-the-art DL frameworks with Tensorflow (but there are others, JuliaDiff, Jax, Caffe,…)

https://github.com/CIA-Oceanix/DLCourse_MOi_2022/blob/main/notebooks/intro_pytorch_learning.ipynb

Introduction to pytorch

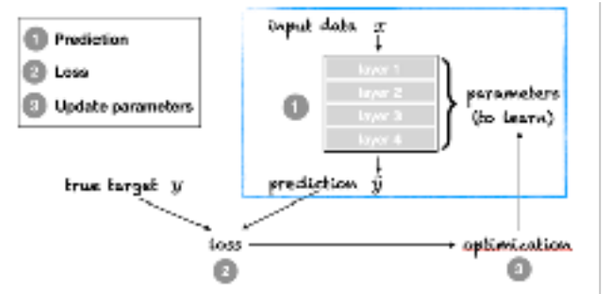
https://github.com/CIA-Oceanix/DLCourse_MOi_2022/blob/main/notebooks/intro_pytorch_learning.ipynb

To be coded:

- Use different optimisers (see https://pytorch.org/docs/stable/optim.html)

- Test different activation functions (see https://pytorch.org/docs/stable/nn.html#non-linear-activations-weighted-sum-nonlinearity)

Guidelines to implement Deep Learning schemes



1. Problem formulation (inputs/outputs)
2. Data collection (cf. supervised vs. non-supervised)
3. Definition of performance metrics
4. Selection of neural architectures (at least 2 models)
5. Selection of a training loss
6. Split dataset into training / validation / test datasets
7. Train the selected models from the training dataset and save the best models onto the validation dataset
8. Benchmark the performance of the trained models onto the test dataset
9. Update/iterate 4-5-6-7-8

Things to know (Introduction to Neural Networks)

- Layer
- Activation function
- Neural network depth
- Backpropagation
- Pytorch